**Microchip Power Line Communications**

**Atmel Binary Sniffer Creation**

**Application Note**

Overview

ATPL Multiprotocol Sniffer uses SQLite to save PRIME/G3 logs but, sometimes, this is not usefull. Sometimes it is more useful to store the raw data as generated by a sniffer device or an embedded sniffer from a Base/Terminal device. This application note describes the binary format defined.

Table of Contents

[1. File Format 4](#_Toc485281061)

[1.2 PRIME Sniffer Frame Format 4](#_Toc485281062)

[1.3 Adding Timestamp information into sniffer frames 5](#_Toc485281063)

[1.3.1 Set Embedded Timestamp Presence Flag 5](#_Toc485281064)

[1.3.2 Write Timestamp 6](#_Toc485281065)

[1.3.3 Add CRC 6](#_Toc485281066)

[1.3.4 Add serial profile escape sequences and begin/end tokens. 7](#_Toc485281067)

[1.3.5 Write to file 7](#_Toc485281068)

[2. Revision History 8](#_Toc485281069)

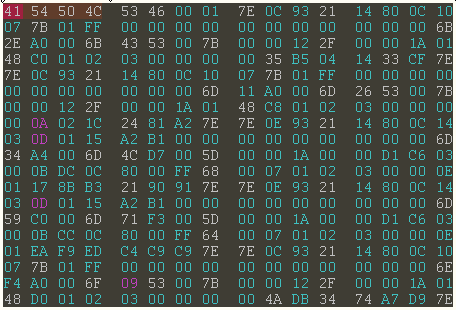
# File Format

The file format is very simple. It contains a small header identifiying the file type and the raw serialization of USI Sniffer frames as received through the serial port. Each sniffer frame is enclosed in *‘7E’* tags as defined in the PRIME Serial communication profile.

In order to know more about USI format, refer to the PRIME Firmware Documentation and PRIME Standard Documentation.

The hearder is simply a magic number of 8 bytes with the value *{0x41, 0x54, 0x50, 0x4C, 0x53, 0x46,0x00,0x01}.* Figure 1-1 show a file dump of a sniffer binary file.

Hexadecimal display of a sniffer file.



File shown above, can be imported into a database in the ATPL Multiprotocol Sniffer using the menu “*File/Import ATPL log”.*

Unfornutatelly, sniffer sources (like embedded sniffer from a Base node) do not have time information. Sometimes it is crucial to have information of when a frame happened.

## PRIME Sniffer Frame Format

The sniffer frame is codified as shown in Figure 1-2 and Figure 1-3. Frames will be modified to add a timestamp.

PRIME Sniffer USI frame format

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **7E**  (1 byte) | MSG LENGTH (10 bits) | PROTOCOL ID **0x13** (6 bits) | MESSAGE DATA | | CRC  (2 bytes) | **7E**  (1 byte) |
| HEADER  (32 bytes) | PDU SNIFFER MSG |

PRIME Sniffer USI header field

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FRA T  1 byte | SNIF F  1 byte | SNIF T  1 byte | MODUL  1 byte | SYM PDU  1 byte | SNR  1 byte | EX SNR  1 byte | CHN  1 byte | CINR  1 byte | BERSOFT  1 byte | BERS MAX  1 byte | 0x00...0x00  8 bytes |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Time Start  4 bytes | Time End  4 bytes | RSSI  2 bytes | 0x00  1 byte | PDU LEN  2 bytes |

Frame fields:

* FRA T: PDU type of the received frame (A, B, BC) (see mode values in file *atpl230.h*).
* SNIF F: Sniffer frame version: 0x14 for current version.
* SNIF T: Sniffer type version: 0x01 for ATPL230.
* MODUL: Modulation scheme of the received frame (see modulation values in file *atpl230.h*). Note that the modulation scheme of frames received in the serial PHY layer is set to 0x0F.
* SYM PDU: Length of the PDU in PHY symbols.
* SNR: PRIME defined measurement of the SNR (from 0 to 7).
* EX SNR: High precision SNR.
* CHN: Channel in which the frame has been received.
* CINR: Minimum Carrier to Interference Noise Ratio.
* BERSOFT: Viterbi soft bit error rate value.
* BERS MAX: Viterbi soft bit error rate maximum value.
* Reserved: eight bytes. Fill with zeroes.
* Time Start/Time End: high precision internal counter to measure length (time) of the PDUs in hundredths of microseconds.
* RSSI: average RSSI in dBuV.
* PDU LEN: Length of the PDU in bytes.

## Adding Timestamp information into sniffer frames

It is possible to embed a timestamp into each sniffer frame. To do so, for each frame received it is needed to:

1. Set Embedded Timestamp presence bit.
2. Write an 8 byte long timestamp containg the miliseconds since epoc of this received frame.
3. Update the CRC at the end of the frame.
4. Add binary escape sequences as defined in the serial communications profile in PRIME.
5. Add head and tail tokens (0x7E).
6. Write to file.

### Set Embedded Timestamp Presence Flag

In the sniffer frame header, modify the *“Sniffer type”* field, setting the 7th most significant bit to one (for example ORing the current value with 0x40). This will tell the ATPL Multiprotocol Sniffer that a timestamp is present.

### Write Timestamp

Timestamp is a 64 bit long, unsigned integer counting the number of miliseconds since EPOC. It must be written in the reserved space in the sniffer header, starting at 12th byte, and with timestamps´ MSB.

A simple way of obtaining the timestamp is:

uint64\_t get\_timestamp\_miliseconds\_since\_epoc()

{

uint64\_t time\_in\_ms;

struct timeval tv;

gettimeofday(&tv,NULL);

time\_in\_ms = ((1000000\* tv.tv\_sec )+ tv.tv\_usec) / 1000;

return time\_in\_ms;

}

### Add CRC

As we have modified the frame, a new CRC must be calculated. USI Sniffer frames use the CRC16 specified in PRIME FW Stack. The implementation of the CRC16 is shown the usi.c file from the Usi Host library:

static const uint16\_t crc16Table[256] =

{

0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,

0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,

0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,

0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,

0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,

0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,

0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,

0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,

0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,

0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,

0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,

0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,

0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,

0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,

0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,

0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,

0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,

0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,

0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,

0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,

0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,

0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,

0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,

0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,

0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,

0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,

0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,

0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,

0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,

0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,

0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,

0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0

};

static uint16\_t \_evalCrc16 (const uint8\_t \*bufPtr, uint16\_t len) // len = 64 -> 2.93 ms.

{

uint16\_t crc;

crc = 0;

while (len--)

crc = (uint16\_t)(crc16Table [(crc >> 8) & 0xff] ^ (crc << 8) ^ (\*bufPtr++ & 0x00ff));

return crc;

}

CRC must be appended at the end of the sniffer frame.

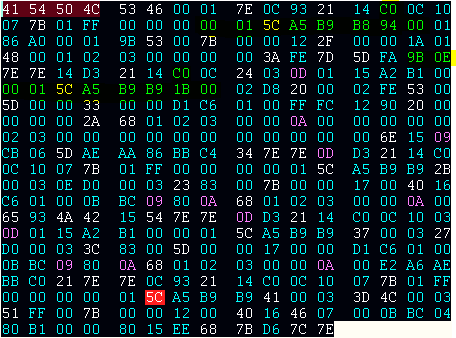
### Add serial profile escape sequences and begin/end tokens.

Now, the frame is modified, it is needed to restore the PRIME serial profile format. This format encapsulates each frame in HDLC frames. Begin and end of a frame is marked with a 0x7E and the frame itself can not contain those values. Escape sequecuences must be added if that happens. PRIME Standard documentation specifies how must be done in the serial communication profile.

### Write to file

When the frame is modified, write it to the file. Figure 1-4 shows a binary sniffer file with the added timestamp. Files with this format can be imported and converted into a SQLite database with the ATPL Multiprotocol Sniffer tool. Timestamp in the imported database will be recorded with the data from the embedded timestamps.

Sniffer file with embedded timestamp.



# Revision History

|  |  |  |
| --- | --- | --- |
| **Doc. Rev.** | **Date** | **Comments** |
| A | 15/06/2017 | Initial document release. |

**\* Do NOT delete the following section break or lines. Doing so, will remove the Atmel copy/disclaimer page and the document's header/footer formatting. This instruction in red will NOT print. It is OKAY to leave it this instruction in the document.**



**Atmel Corporation** 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 **│** [www.atmel.com](http://www.atmel.com)

© 2017 Atmel Corporation. / Rev. Microchip-43102A-MCPL-G3 Manager-UserGuide\_24-May-17.

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries.

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death (“Safety-Critical Applications”) without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.